Low-Rank Approximation of Weighted Tree Automata

Guillaume Rabusseau
Aix-Marseille University

Borja Balle
Lancaster University

Shay B. Cohen
University of Edinburgh

Abstract

We describe a technique to minimize weighted tree automata (WTA), a powerful formalism that subsumes probabilistic context-free grammars (PCFGs) and latent-variable PCFGs. Our method relies on a singular value decomposition of the underlying Hankel matrix defined by the WTA. Our main theoretical result is an efficient algorithm for computing the SVD of an infinite Hankel matrix implicitly represented as a WTA. We evaluate our method on real-world data originating in newswire treebank. We show that our approach achieves lower perplexity than previous methods for PCFG minimization, and also is much more stable due to the absence of local optima.

1 Introduction

Probabilistic context-free grammars (PCFG) provide a powerful statistical formalism for modeling important phenomena occurring in natural language. In fact, learning and parsing algorithms for PCFG are now standard tools in natural language processing pipelines. Most of these algorithms can be naturally extended to the superclass of weighted context-free grammars (WCFG), and closely related models like weighted tree automata (WTA) and latent probabilistic context-free grammars (LPCFG). The complexity of these algorithms depends on the size of the grammar/automaton, typically controlled by the number of rules/states. Being able to control this complexity is essential in operations like parsing, which is typically executed every time the model is used to make a prediction. In this paper we present an algorithm that given a WTA with n states and a target number of states $\hat{n} < n$, returns a WTA with $\hat{n}$ states that is a good approximation of the original automaton. This can be interpreted as a low-rank approximation method for WTA through the direct connection between number of states of a WTA and the rank of its associated Hankel matrix. This opens the door to reducing the complexity of algorithms working with WTA at the price of incurring a small, controlled amount of error in the output of such algorithms. For example, the complexity for parsing a tree t using a WTA is cubic in the number of states (Maletti and Satta, 2009), thus reducing the number of states can lead to a significant speed-up for inference time with a minimized model.

Our techniques are inspired by recent developments in spectral learning algorithms for different classes of models on sequences (Hsu et al., 2012; Bailly et al., 2009; Boots et al., 2011; Balle et al., 2014) and trees (Bailly et al., 2010; Cohen et al., 2014), and subsequent investigations into low-rank spectral learning for predictive state representations (Kulesza et al., 2014, 2015) and approximate minimization of weighted automata (Balle et al., 2015). In spectral learning algorithms, data is used to reconstruct a finite block of a Hankel matrix and an SVD of such matrix then reveals a low-dimensional space where a linear regression recovers the parameters of the model. In contrast, our approach computes the SVD of the *infinite* Hankel matrix associated with a WTA, and then uses it to obtain a low-rank approximation to the initial WTA. Our main result is an efficient algorithm for computing this singular value decomposition by operating directly on the WTA representation of the Hankel matrix; that is, without the need to explicitly represent this infinite matrix at any point. Section 2 presents the main ideas underlying our approach and an efficient algorithmic implementation of these ideas is discussed in Section 3. Proofs of all results stated in the paper can be found in the supplementary materials.

The idea of speeding up parsing with (L)PCFG by approximating the original model with a smaller one was recently studied in (Cohen and Collins, 2012; Cohen et al., 2013a), where a tensor decomposition technique was used in order to obtain the minimized model. We

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

compare that approach to ours in the experiments presented in Section 4, where both techniques are used to compute approximations to a grammar learned from a corpus of real linguistic data. It was observed in (Cohen and Collins, 2012; Cohen et al., 2013a) that a side-effect of reducing the size of a grammar learned from data was a slight improvement in parsing performance. The number of parameters in the approximate models is smaller, and as such, generalization improves. We show in our experimental section that our minimization algorithms have the same effect in certain parsing scenarios. In addition, our approach yields models which give lower perplexity on an unseen set of sentences, and provides a better approximation to the original model in terms of ℓ_2 distance. It is important to remark that in contrast with the tensor decompositions in (Cohen and Collins, 2012; Cohen et al., 2013a) which are susceptible to local optima problems, our approach resembles a power-method approach to SVD, which yields efficient globally convergent algorithms. Overall, we observe in our experiments that this renders a more stable minimization method than the one using tensor decompositions.

1.1 Notation

For an integer n, we write $[n] = \{1, \ldots, n\}$. We use lower case bold letters (or symbols) for vectors (e.g. $\mathbf{v} \in \mathbb{R}^{d_1}$), upper case bold letters for matrices (e.g. $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$) and bold calligraphic letters for third order tensors (e.g. $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$). Unless explicitly stated, vectors are by default column vectors. The identity matrix will be written as $\mathbf{I}$. Given $i_1 \in [d_1], i_2 \in [d_2], i_3 \in [d_3]$ we use $\mathbf{v}(i_1)$, $\mathbf{M}(i_1, i_2)$, and $\boldsymbol{\mathcal{T}}(i_1, i_2, i_3)$ to denote the corresponding entries. The ith row (resp. column) of a matrix $\mathbf{M}$ will be noted $\mathbf{M}(i, :)$ (resp. $\mathbf{M}(:, i)$). This notation is extended to slices across the three modes of a tensor in the straightforward way. If $\mathbf{v} \in \mathbb{R}^{d_1}$ and $\mathbf{v}' \in \mathbb{R}^{d_2}$, we use $\mathbf{v} \otimes \mathbf{v}' \in \mathbb{R}^{d_1 \cdot d_2}$ to denote the Kronecker product between vectors, and its straightforward extension to matrices and tensors. Given a matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ we use $\operatorname{vec}(\mathbf{M}) \in \mathbb{R}^{d_1 \cdot d_2}$ to denote the column vector obtained by concatenating the columns of $\mathbf{M}$. Given a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and matrices $\mathbf{M}_i \in \mathbb{R}^{d_i \times d'_i}$ for $i \in [3]$, we define a tensor $\boldsymbol{\mathcal{T}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3) \in \mathbb{R}^{d'_1 \times d'_2 \times d'_3}$ whose entries are given by

$$\boldsymbol{\mathcal{T}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3)(i_1, i_2, i_3) = \sum_{j_1, j_2, j_3} \boldsymbol{\mathcal{T}}(j_1, j_2, j_3) \mathbf{M}_1(j_1, i_1) \mathbf{M}_2(j_2, i_2) \mathbf{M}_3(j_3, i_3) \enspace .$$

This operation corresponds to contracting $\boldsymbol{\mathcal{T}}$ with $\mathbf{M}_i$ across the ith mode of the tensor for each i.

2 Approximate Minimization of WTA and SVD of Hankel Matrices

In this section we present the first contribution of the paper. Namely, the existence of a canonical form for weighted tree automata inducing the singular value decomposition of the infinite Hankel matrix associated with the automaton. We start by recalling several definitions and well-known facts about WTA that will be used in the rest of the paper. Then we proceed to establish the existence of the canonical form, which we call the *singular value tree automaton*. Finally we indicate how removing the states in this canonical form that correspond to the smallest singular values of the Hankel matrix leads to an effective procedure for model reduction in WTA.

2.1 Weighted Tree Automata

Let Σ be a finite alphabet. We use $\Sigma^\star$ to denote the set of all finite strings with symbols in Σ with λ denoting the empty string. We write $|x|$ to denote the length of a string $x \in \Sigma^\star$. The number of occurrences of a symbol $\sigma \in \Sigma$ in a string $x \in \Sigma^\star$ is denoted by $|x|_\sigma$.

We now introduce notation for describing the trees generated by a tree automaton; see Figure 1 for some illustrative examples. The set of all *rooted full binary trees* with leafs in Σ is the smallest set $\mathfrak{T}_\Sigma$ such that $\Sigma \subset \mathfrak{T}_\Sigma$ and $(t_1, t_2) \in \mathfrak{T}_\Sigma$ for any $t_1, t_2 \in \mathfrak{T}_\Sigma$. We shall just write $\mathfrak{T}$ when the alphabet Σ is clear from the context. The *size* of a tree $t \in \mathfrak{T}$ is denoted by $\operatorname{size}(t)$ and defined recursively by $\operatorname{size}(\sigma) = 0$ for $\sigma \in \Sigma$, and $\operatorname{size}((t_1, t_2)) = \operatorname{size}(t_1) + \operatorname{size}(t_2) + 1$; that is, the number of internal nodes in the tree. The *depth* of a tree $t \in \mathfrak{T}$ is denoted by $\operatorname{depth}(t)$ and defined recursively by $\operatorname{depth}(\sigma) = 0$ for $\sigma \in \Sigma$, and $\operatorname{depth}((t_1, t_2)) = \max\{\operatorname{depth}(t_1), \operatorname{depth}(t_2)\} + 1$; that is, the distance from the root of the tree to the farthest leaf. The *yield* of a tree $t \in \mathfrak{T}$ is a string $\langle t \rangle \in \Sigma^*$ defined as the left-to-right concatenation of the symbols in the leafs of t, and can be recursively defined by $\langle \sigma \rangle = \sigma$, and $\langle (t_1, t_2) \rangle = \langle t_1 \rangle \cdot \langle t_2 \rangle$. The total number of nodes (internal plus leafs) of a tree t is denoted by $|t|$ and satisfies $|t| = \operatorname{size}(t) + |\langle t \rangle|$.

Let $\Sigma' = \Sigma \cup \{*\}$, where $*$ is a symbol *not* in Σ. The set of *rooted full binary context trees* is the set $\mathfrak{C}_\Sigma = \{c \in \mathfrak{T}_{\Sigma'} \mid |\langle c \rangle|_* = 1\}$; that is, a context $c \in \mathfrak{C}_\Sigma$ is a tree in $\mathfrak{T}_{\Sigma'}$ in which the symbol $*$ occurs exactly in one leaf. Note that because given a context $c = (t_1, t_2) \in \mathfrak{C}_\Sigma$ with $t_1, t_2 \in \mathfrak{T}_{\Sigma'}$ the symbol $*$ can only appear in one of the t_1 and t_2, we must actually have $c = (c', t)$ or $c = (t, c')$ with $c' \in \mathfrak{C}_\Sigma$ and $t \in \mathfrak{T}_\Sigma$. The *drop* of a context $c \in \mathfrak{C}$ is the distance between the root and the leaf labeled with $*$ in c, which can be defined recursively as

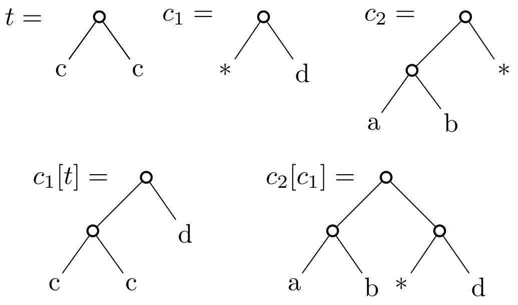

Figure 1: Examples of trees $(t, c_1[t] \in \mathfrak{T}_\Sigma)$ and contexts $(c_1, c_2, c_2[c_1] \in \mathfrak{C}_\Sigma)$ on the alphabet $\Sigma = \{a, b, c, d\}$. In our notation: $c_1[t] = ((c,c),d)$, $\mathrm{size}(c_1[t]) = 2$, $\mathrm{depth}(c_1[t]) = 2$, $\langle t \rangle = cc$, $\mathrm{drop}(c_2[c_1]) = 2$

$\mathrm{drop}(*) = 0$, $\mathrm{drop}((c,t)) = \mathrm{drop}((t,c)) = \mathrm{drop}(c) + 1$.

We usually think as the leaf with the symbol $*$ in a context as a placeholder where the root of another tree or another context can be inserted. Accordingly, given $t \in \mathfrak{T}$ and $c \in \mathfrak{C}$, we can define $c[t] \in \mathfrak{T}$ as the tree obtained by replacing the occurrence of $*$ in c with t. Similarly, given $c, c' \in \mathfrak{C}$ we can obtain a new context tree $c[c']$ by replacing the occurrence of $*$ in c with c'.

A *weighted tree automaton* (WTA) over Σ is a tuple $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}_{\sigma \in \Sigma} \rangle$, where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the vector of *initial weights*, $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times n \times n}$ is the tensor of *transition weights*, and $\boldsymbol{\omega}_\sigma \in \mathbb{R}^n$ is the vector of *terminal weights* associated with $\sigma \in \Sigma$. The dimension n is the number of states of the automaton, which we shall sometimes denote by $|A|$. A WTA $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\} \rangle$ *computes* a function $f_A : \mathfrak{T}_\Sigma \to \mathbb{R}$ assigning to each tree $t \in \mathfrak{T}$ the number computed as $f_A(t) = \boldsymbol{\alpha}^\top \boldsymbol{\omega}_A(t)$, where $\boldsymbol{\omega}_A(t) \in \mathbb{R}^n$ is obtained recursively as $\boldsymbol{\omega}_A(\sigma) = \boldsymbol{\omega}_\sigma$, and $\boldsymbol{\omega}_A((t_1, t_2)) = \boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \boldsymbol{\omega}_A(t_2))$ — note the matching of dimensions in this last expression since contracting a third order tensor with a matrix in the first mode and vectors in the second and third mode yields a vector. In many cases we shall just write $\boldsymbol{\omega}(t)$ when the automaton A is clear from the context.

Although WTA are traditionally studied as recognizers for (weighted) regular tree languages, the computation performed by a WTA is closely related to several models typically used in machine learning, including PCFG and recursive tensor neural networks. The connection with PCFG can be obtained by noting that mapping each non-terminal symbol in a PCFG to a different state of a WTA one can obtain an automaton A with the property that if t is a tree with yield $\langle t \rangle = w$, then the value of $f_A(t)$ equals the sum of the probabilities of all derivation trees for w in the original grammar having the same topology as t (see also Section 4.1). Thus, WTA can compute the same weighted context-free languages as WCFG. The connection with the recursive tensor neural networks (RTNN) introduced in (Socher et al., 2013) follows from observing that the computation of a WTA has the same bottom-up computational structure as a RTNN without the non-linearities. The values of the n components of a leaf vector $\boldsymbol{\omega}_\sigma$ correspond to the values of n features representing symbol $\sigma \in \Sigma$. The computation performed by a WTA processes a tree from the bottom up: whenever a tree of the form $t = (t_1, t_2)$ is encountered, it first processes the subtrees t_1 and t_2 to obtain the feature vectors $\boldsymbol{\omega}(t_1)$ and $\boldsymbol{\omega}(t_2)$, and then computes a new feature vector for t as $\boldsymbol{\omega}_A(t) = \boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \boldsymbol{\omega}_A(t_2))$. At the top level the computation ends by producing the scalar $f_A(t) = \boldsymbol{\alpha}^\top \boldsymbol{\omega}(t)$ corresponding to the inner product of the feature representation of t with the vector $\boldsymbol{\alpha}$.

While WTA are usually defined over arbitrary ranked trees, only considering binary trees does not lead to any loss of generality since WTA on ranked trees are equivalent to WTA on binary trees (see Bailly et al. (2010) for references). Additionally, one could consider binary trees where each internal node is decorated with labeled from a finite set, which leads to the definition of WTA with multiple transition tensors. Our results can be extended to this case without much effort, but we state them just for WTA with only one transition tensor to keep the notation manageable.

An important observation is that there exist more than one WTA computing the same function — in fact, there exist infinitely many. An important construction along these lines is the *conjugate* of a WTA A with n states by an invertible matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$. If $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\} \rangle$, its conjugate by $\mathbf{Q}$ is $A^{\mathbf{Q}} = \langle \mathbf{Q}^\top \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}(\mathbf{Q}^{-\top}, \mathbf{Q}, \mathbf{Q}), \{\mathbf{Q}^{-1} \boldsymbol{\omega}_\sigma\} \rangle$, where $\mathbf{Q}^{-\top}$ denotes the inverse of $\mathbf{Q}^\top$. To show that $f_A = f_{A^{\mathbf{Q}}}$ one applies an induction argument on $\mathrm{depth}(t)$ to show that for every $t \in \mathfrak{T}$ one has $\boldsymbol{\omega}_{A^{\mathbf{Q}}}(t) = \mathbf{Q}^{-1} \boldsymbol{\omega}_A(t)$. The claim is obvious for trees of zero depth $\sigma \in \Sigma$, and for $t = (t_1, t_2)$ one has

$$\begin{aligned}
\boldsymbol{\omega}_{A^{\mathbf{Q}}}((t_1, t_2)) &= (\boldsymbol{\mathcal{T}}(\mathbf{Q}^{-\top}, \mathbf{Q}, \mathbf{Q}))(\mathbf{I}, \boldsymbol{\omega}_{A^{\mathbf{Q}}}(t_1), \boldsymbol{\omega}_{A^{\mathbf{Q}}}(t_2)) \\
&= (\boldsymbol{\mathcal{T}}(\mathbf{Q}^{-\top}, \mathbf{Q}, \mathbf{Q}))(\mathbf{I}, \mathbf{Q}^{-1} \boldsymbol{\omega}_A(t_1), \mathbf{Q}^{-1} \boldsymbol{\omega}_A(t_2)) \\
&= \boldsymbol{\mathcal{T}}(\mathbf{Q}^{-\top}, \boldsymbol{\omega}_A(t_1), \boldsymbol{\omega}_A(t_2)) \\
&= \mathbf{Q}^{-1} \boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \boldsymbol{\omega}_A(t_2)) \ ,
\end{aligned}$$

where we just used some simple rules of tensor algebra.

An arbitrary function $f : \mathfrak{T} \to \mathbb{R}$ is called *rational* if there exists a WTA A such that $f = f_A$. The number of states of the smallest such WTA is the *rank* of f — we shall set $\mathit{rank}(f) = \infty$ if f is not rational. A WTA A with $f_A = f$ and $|A| = \mathit{rank}(f)$ is called *minimal*. Given any $f : \mathfrak{T} \to \mathbb{R}$ we define its *Hankel matrix* as the infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\mathfrak{C} \times \mathfrak{T}}$ with rows indexed by contexts, columns indexed by trees, and whose entries are given by $\mathbf{H}_f(c,t) = f(c[t])$. Note that given a tree $t' \in \mathfrak{T}$ there are exactly $|t'|$ different ways of splitting $t' = c[t]$ with $c \in \mathfrak{C}$ and $t \in \mathfrak{T}$. This implies that $\mathbf{H}_f$ is a highly redundant representation for f, and it turns out that this redundancy is the key to proving the following fundamental result about rational tree functions.

Theorem 1 ((Bozapalidis and Louscou-Bozapalidou, 1983))**.** *For any $f : \mathfrak{T} \to \mathbb{R}$ we have $rank(f) = rank(\mathbf{H}_f)$.*

2.2 Rank Factorizations of Hankel Matrices

The theorem above can be rephrased as saying that the rank of $\mathbf{H}_f$ is finite if and only if f is rational. When the rank of $\mathbf{H}_f$ is indeed finite — say $rank(\mathbf{H}_f) = n$ — one can find two rank n matrices $\mathbf{P} \in \mathbb{R}^{\mathfrak{C} \times n}$, $\mathbf{S} \in \mathbb{R}^{n \times \mathfrak{T}}$ such that $\mathbf{H}_f = \mathbf{PS}$. In this case we say that $\mathbf{P}$ and $\mathbf{S}$ give a *rank factorization* of $\mathbf{H}_f$. We shall now refine Theorem 1 by showing that when f is rational, the set of all possible rank factorizations of $\mathbf{H}_f$ is in direct correspondence with the set of minimal WTA computing f.

The first step is to show that any minimal WTA $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}\rangle$ computing f induces a rank factorization $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$. We build $\mathbf{S}_A \in \mathbb{R}^{n \times \mathfrak{T}}$ by setting the column corresponding to a tree t to $\mathbf{S}_A(:,t) = \boldsymbol{\omega}_A(t)$. In order to define $\mathbf{P}_A$ we need to introduce a new mapping $\Xi_A : \mathfrak{C} \to \mathbb{R}^{n \times n}$ assigning a matrix to every context as follows: $\Xi_A(*) = \mathbf{I}$, $\Xi_A((c,t)) = \boldsymbol{\mathcal{T}}(\mathbf{I}, \Xi_A(c), \boldsymbol{\omega}_A(t))$, and $\Xi_A((t,c)) = \boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\omega}_A(t), \Xi_A(c))$. If we now define $\boldsymbol{\alpha}_A : \mathfrak{C} \to \mathbb{R}^n$ as $\boldsymbol{\alpha}_A(c)^\top = \boldsymbol{\alpha}^\top \Xi_A(c)$, we can set the row of $\mathbf{P}_A$ corresponding to c to be $\mathbf{P}_A(c,:) = \boldsymbol{\alpha}_A(c)^\top$. With these definitions one can easily show by induction on drop(c) that $\Xi_A(c)\boldsymbol{\omega}_A(t) = \boldsymbol{\omega}_A(c[t])$ for any $c \in \mathfrak{C}$ and $t \in \mathfrak{T}$. Then it is immediate to check that $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$:

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{P}_A(c,i)\mathbf{S}_A(i,t) &= \boldsymbol{\alpha}_A(c)^\top \boldsymbol{\omega}_A(t) = \boldsymbol{\alpha}^\top \Xi_A(c)\boldsymbol{\omega}_A(t) \\ &= \boldsymbol{\alpha}^\top \boldsymbol{\omega}_A(c[t]) = f_A(c[t]) \\ &= \mathbf{H}_f(c,t) \ . \qquad (1)\end{aligned}$$

As before, we shall sometimes just write $\Xi(c)$ and $\boldsymbol{\alpha}(c)$ when A is clear from the context. We can now state the main result of this section, which generalizes similar results in (Balle et al., 2015, 2014) for weighted automata on strings.

Theorem 2. *Let $f : \mathfrak{T} \to \mathbb{R}$ be rational. If $\mathbf{H}_f = \mathbf{PS}$ is a rank factorization, then there exists a minimal WTA A computing f such that $\mathbf{P}_A = \mathbf{P}$ and $\mathbf{S}_A = \mathbf{S}$.*

Proof. See supplementary material. □

2.3 Approximate Minimization with the Singular Value Tree Automaton

Equation (1) can be interpreted as saying that given a fixed factorization $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$, the value $f_A(c[t])$ is given by the inner product $\sum_i \boldsymbol{\alpha}_A(c)_i \boldsymbol{\omega}_A(t)_i$. Thus, $\boldsymbol{\alpha}_A(c)_i$ and $\boldsymbol{\omega}_A(t)_i$ quantify the influence of state i in the computation of $f_A(c[t])$, and by extension one can use $\|\mathbf{P}_A(:,i)\|$ and $\|\mathbf{S}_A(i,:)\|$ to measure the overall influence of state i in f_A. Since our goal is to approximate a given WTA by a smaller WTA obtained by removing some states in the original one, we shall proceed by removing those states with overall less influence on the computation of f. But because there are infinitely many WTA computing f, we need to first fix a particular representation for f before we can remove the less influential states. In particular, we seek a representation where each state is decoupled as much as possible from each other state, and where there is a clear ranking of states in terms of overall influence. It turns out all this can be achieved by a canonical form for WTA we call the singular value tree automaton, which provides an implicit representation for the SVD of $\mathbf{H}_f$. We now show conditions for the existence of such canonical form, and in the next section we develop an algorithm for computing it efficiently.

Suppose $f : \mathfrak{T} \to \mathbb{R}$ is a rank n rational function such that its Hankel matrix admits a reduced singular value decomposition $\mathbf{H}_f = \mathbf{UDV}^\top$. Then we have that $\mathbf{P} = \mathbf{UD}^{1/2}$ and $\mathbf{S} = \mathbf{D}^{1/2}\mathbf{V}^\top$ is a rank decomposition for $\mathbf{H}_f$, and by Theorem 2 there exists some minimal WTA A with $f_A = f$, $\mathbf{P}_A = \mathbf{UD}^{1/2}$ and $\mathbf{S}_A = \mathbf{D}^{1/2}\mathbf{V}^\top$. We call such an A a *singular value tree automaton* (SVTA) for f. However, these are not defined for every rational function f, because the fact that columns of $\mathbf{U}$ and $\mathbf{V}$ must be unitary vectors (i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$) imposes some restrictions on which infinite Hankel matrices $\mathbf{H}_f$ admit an SVD — this phenomenon is related to the distinction between compact and non-compact operators in functional analysis. Our next theorem gives a sufficient condition for the existence of an SVD of $\mathbf{H}_f$.

We say that a function $f : \mathfrak{T} \to \mathbb{R}$ is *strongly convergent* if the series $\sum_{t \in \mathfrak{T}} |t||f(t)|$ converges. To see the intuitive meaning of this condition, assume that f is a probability distribution over trees in $\mathfrak{T}$. In this case, strong convergence is equivalent to saying that the expected size of trees generated from the distribution f is finite. It turns out strong convergence of f is a sufficient condition to guarantee the existence of an SVD for $\mathbf{H}_f$.

Theorem 3. *If $f : \mathfrak{T}_\Sigma \to \mathbb{R}$ is rational and strongly convergent, then $\mathbf{H}_f$ admits a singular value decomposition.*

Proof. See supplementary material. □

Together, Theorems 2 and 3 imply that every rational strongly convergent $f : \mathfrak{T} \to \mathbb{R}$ can be represented by an SVTA A. If $rank(f) = n$, then A has n states and for every $i \in [n]$ the ith state contributes to $\mathbf{H}_f$ by generating the ith left and right singular vectors weighted by $\sqrt{\mathfrak{s}_i}$, where $\mathfrak{s}_i = \mathbf{D}(i,i)$ is the ith singular value. Thus, if we desire to obtain a good approximation $\hat{f}$ to f with $\hat{n}$ states, we can take the WTA $\hat{A}$ obtained by removing the last $n - \hat{n}$ states from A, which corresponds to removing from f the contribution of the smallest singular values of $\mathbf{H}_f$. We call such $\hat{A}$ an *SVTA truncation.* Given an SVTA $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}\rangle$ and $\mathbf{\Pi} = [\mathbf{I} \mid \mathbf{0}] \in \mathbb{R}^{\hat{n}\times n}$, the SVTA truncation to $\hat{n}$ states can be written as

$$\hat{A} = \langle \mathbf{\Pi}\boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}(\mathbf{\Pi}^\top, \mathbf{\Pi}^\top, \mathbf{\Pi}^\top), \{\mathbf{\Pi}\boldsymbol{\omega}_\sigma\}\rangle \ .$$

Intuitively, the states associated with the smaller singular values are the ones with the less influence on the Hankel matrix, thus they should also be the states having the less effect on the computation of the SVTA. The following theorem support this intuition by showing a fundamental relation between the singular values of the Hankel matrix of a rational function f and the parameters of the SVTA computing it.

Theorem 4. *Let $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}_{\sigma\in\Sigma}\rangle$ be a SVTA with n states realizing a function f and let $\mathfrak{s}_1 \geq \mathfrak{s}_2 \geq \cdots \geq \mathfrak{s}_n$ be the singular values of the Hankel matrix $\mathbf{H}_f$.*

Then, for any $t \in \mathfrak{T}$, $c \in \mathfrak{C}$ and $i,j,k \in [n]$ the following hold:

- $|\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i}$,
- $|\boldsymbol{\alpha}(c)_i| \leq \sqrt{\mathfrak{s}_i}$ *, and*
- $|\boldsymbol{\mathcal{T}}(i,j,k)| \leq \min\{\frac{\sqrt{\mathfrak{s}_i}}{\sqrt{\mathfrak{s}_j}\sqrt{\mathfrak{s}_k}}, \frac{\sqrt{\mathfrak{s}_j}}{\sqrt{\mathfrak{s}_i}\sqrt{\mathfrak{s}_k}}, \frac{\sqrt{\mathfrak{s}_k}}{\sqrt{\mathfrak{s}_i}\sqrt{\mathfrak{s}_j}}\}$.

Proof. See supplementary material. □

Two important properties of SVTAs follow from this proposition. First, the fact that $|\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i}$ implies that the weights associated with states corresponding to small singular values are small. Second, this proposition gives us some intuition on how the states of an SVTA interact with each other. To see this, let $\mathbf{M} = \boldsymbol{\mathcal{T}}(\boldsymbol{\alpha}, \mathbf{I}, \mathbf{I})$ and remark that for a tree $t = (t_1, t_2) \in \mathfrak{T}$ we have $f(t) = \boldsymbol{\omega}(t_1)^\top \mathbf{M}\boldsymbol{\omega}(t_2)$. Using the previous theorem one can show that

$$|\mathbf{M}(i,j)| \leq n\sqrt{\frac{\min\{\mathfrak{s}_i, \mathfrak{s}_j\}}{\max\{\mathfrak{s}_i, \mathfrak{s}_j\}}} \ ,$$

which tells us that two states corresponding to singular values far away from each other have very little interaction in the computations of the automata.

3 Computing the Singular Value WTA

Previous section shows that in order to compute an approximation to a strongly convergent rational function $f : \mathfrak{T} \to \mathbb{R}$ one can proceed by truncating its SVTA. However, the only obvious way to obtain such SVTA is by computing the SVD of the infinite matrix $\mathbf{H}_f$. In this section we show that if we are given an arbitrary minimal WTA A for f, then we can transform A into the corresponding SVTA efficiently.[1] In other words, given a representation of $\mathbf{H}_f$ as a WTA, we can compute its SVD *without* the need to operate on infinite matrices. The key observation is to reduce the computation of the SVD of $\mathbf{H}_f$ to the computation of spectral properties of the Gram matrices $\mathbf{G}_\mathfrak{C} = \mathbf{P}^\top\mathbf{P}$ and $\mathbf{G}_\mathfrak{T} = \mathbf{S}\mathbf{S}^\top$ associated with the rank factorization $\mathbf{H}_f = \mathbf{PS}$ induced by some minimal WTA computing f. In the case of weighted automata on strings, (Balle et al., 2015) recently showed a polynomial time algorithm for computing the Gram matrices of a *string* Hankel matrix by solving a system of linear equations. Unfortunately, extending their approach to the tree case requires obtaining a closed-form solution to a system of quadratic equations, which in general does not exist. Thus, we shall resort to a different algorithmic technique and show that $\mathbf{G}_\mathfrak{C}$ and $\mathbf{G}_\mathfrak{T}$ can be obtained as fixed points of a certain non-linear operator. This yields the iterative algorithm presented in Algorithm 2 which converges exponentially fast as shown in Theorem 6. The overall procedure to transform a WTA into the corresponding SVTA is presented in Algorithm 1.

We start with a simple linear algebra result showing exactly how to relate the eigendecompositions of $\mathbf{G}_\mathfrak{C}$ and $\mathbf{G}_\mathfrak{T}$ with the SVD of $\mathbf{H}_f$.

Lemma 1. *Let $f : \mathfrak{T} \to \mathbb{R}$ be a rational function such that its Hankel matrix $\mathbf{H}_f$ admits an SVD. Suppose $\mathbf{H}_f = \mathbf{PS}$ is a rank factorization. Then the following hold:*

1. $\mathbf{G}_\mathfrak{C} = \mathbf{P}^\top\mathbf{P}$ *and* $\mathbf{G}_\mathfrak{T} = \mathbf{S}\mathbf{S}^\top$ *are finite symmetric positive definite matrices with eigendecompositions* $\mathbf{G}_\mathfrak{C} = \mathbf{V}_\mathfrak{C}\mathbf{D}_\mathfrak{C}\mathbf{V}_\mathfrak{C}^\top$ *and* $\mathbf{G}_\mathfrak{T} = \mathbf{V}_\mathfrak{T}\mathbf{D}_\mathfrak{T}\mathbf{V}_\mathfrak{T}^\top$.
2. *If* $\mathbf{M} = \mathbf{D}_\mathfrak{C}^{1/2}\mathbf{V}_\mathfrak{C}^\top\mathbf{V}_\mathfrak{T}\mathbf{D}_\mathfrak{T}^{1/2}$ *has SVD* $\mathbf{M} = \tilde{\mathbf{U}}\mathbf{D}\tilde{\mathbf{V}}^\top$, *then* $\mathbf{H}_f = \mathbf{UDV}^\top$ *is an SVD, where* $\mathbf{U} = \mathbf{P}\mathbf{V}_\mathfrak{C}\mathbf{D}_\mathfrak{C}^{-1/2}\tilde{\mathbf{U}}$, *and* $\mathbf{V}^\top = \tilde{\mathbf{V}}^\top\mathbf{D}_\mathfrak{T}^{-1/2}\mathbf{V}_\mathfrak{T}^\top\mathbf{S}$.

[1] If the WTA given to the algorithm is not minimal, a pre-processing step can be used to minimize the input using the algorithm from (Kiefer et al., 2015).

Proof. The proof follows along the same lines as that of (Balle et al., 2015, Lemma 7). □

Putting together Lemma 1 and the proof of Theorem 2 we see that given a minimal WTA computing a strongly convergent rational function, Algorithm 1 below will compute the corresponding SVTA. Note the algorithm depends on a procedure for computing the Gram matrices $\mathbf{G}_\mathfrak{T}$ and $\mathbf{G}_\mathfrak{C}$. In the remaining of this section we present one of our main results: a linearly convergent iterative algorithm for computing these matrices.

Algorithm 1 `ComputeSVTA`

```
Input: A strongly convergent minimal WTA $A$
Output: The corresponding SVTA
  $\mathbf{G}_\mathfrak{C}, \mathbf{G}_\mathfrak{T} \leftarrow$ GramMatrices($A$)
  Let $\mathbf{G}_\mathfrak{T} = \mathbf{V}_\mathfrak{T}\mathbf{D}_\mathfrak{T}\mathbf{V}_\mathfrak{T}^\top$ and $\mathbf{G}_\mathfrak{C} = \mathbf{V}_\mathfrak{C}\mathbf{D}_\mathfrak{C}\mathbf{V}_\mathfrak{C}^\top$ be the eigendecompositions of $\mathbf{G}_\mathfrak{T}$ and $\mathbf{G}_\mathfrak{C}$
  Let $\mathbf{M} = \mathbf{D}_\mathfrak{C}^{1/2}\mathbf{V}_\mathfrak{C}^\top\mathbf{V}_\mathfrak{T}\mathbf{D}_\mathfrak{T}^{1/2}$ and let $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ be the singular value decomposition of $\mathbf{M}$
  Let $\mathbf{Q} = \mathbf{V}_\mathfrak{C}\mathbf{D}_\mathfrak{C}^{-1/2}\mathbf{U}\mathbf{D}^{1/2}$
  return $A^\mathbf{Q}$
```

Let $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}\rangle$ be a strongly convergent WTA of dimension n computing a function f. We now show how the Gram matrix $\mathbf{G}_\mathfrak{T}$ can be approximated using a simple iterative scheme. Let $A^\otimes = \langle \boldsymbol{\alpha}^\otimes, \boldsymbol{\mathcal{T}}^\otimes, \{\boldsymbol{\omega}_\sigma^\otimes\}\rangle$ where $\boldsymbol{\alpha}^\otimes = \boldsymbol{\alpha} \otimes \boldsymbol{\alpha}$, $\boldsymbol{\mathcal{T}}^\otimes = \boldsymbol{\mathcal{T}} \otimes \boldsymbol{\mathcal{T}} \in \mathbb{R}^{n^2 \times n^2 \times n^2}$ and $\boldsymbol{\omega}_\sigma^\otimes = \boldsymbol{\omega}_\sigma \otimes \boldsymbol{\omega}_\sigma$ for all $\sigma \in \Sigma$. It is shown in (Berstel and Reutenauer, 1982) that $A^\otimes$ computes the function $f_{A^\otimes}(t) = f(t)^2$. Note we have $\mathbf{G}_\mathfrak{T} = \mathbf{S}\mathbf{S}^\top = \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top$, hence $\mathbf{s} \triangleq \mathrm{vec}(\mathbf{G}_\mathfrak{T}) = \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}^\otimes(t)$ since $\boldsymbol{\omega}^\otimes(t) = \mathrm{vec}(\boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top)$. Thus, computing the Gram matrix $\mathbf{G}_\mathfrak{T}$ boils down to computing the vector $\mathbf{s}$. The following theorem shows that this can be done by repeated applications of a non-linear operator until convergence to a fixed point.

Theorem 5. *Let $F : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ be the mapping defined by $F(\mathbf{v}) = \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{v}, \mathbf{v}) + \sum_{\sigma\in\Sigma} \boldsymbol{\omega}_\sigma^\otimes$. Then the following hold:*

(i) *$\mathbf{s}$ is a fixed-point of F; i.e. $F(\mathbf{s}) = \mathbf{s}$.*

(ii) *$\mathbf{0}$ is in the basin of attraction of $\mathbf{s}$; i.e. $\lim_{k\to\infty} F^k(\mathbf{0}) = \mathbf{s}$.*

(iii) *The iteration defined by $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{s}_{k+1} = F(\mathbf{s}_k)$ converges linearly to $\mathbf{s}$; i.e. there exists $0 < \rho < 1$ such that $\|\mathbf{s}_k - \mathbf{s}\|_2 \leq \mathcal{O}(\rho^k)$.*

Proof. See supplementary material. □

Though we could derive a similar iterative algorithm for computing $\mathbf{G}_\mathfrak{C}$, it turns out that knowledge of $\mathbf{s} = \mathrm{vec}(\mathbf{G}_\mathfrak{T})$ provides an alternative, more efficient procedure for obtaining $\mathbf{G}_\mathfrak{C}$. Like before, we have $\mathbf{G}_\mathfrak{C} = \mathbf{P}^\top\mathbf{P} = \sum_{c\in\mathfrak{C}} \boldsymbol{\alpha}(c)\boldsymbol{\alpha}(c)^\top$ and $\boldsymbol{\alpha}^\otimes(c) = \boldsymbol{\alpha}(c) \otimes \boldsymbol{\alpha}(c)$ for all $c \in \mathfrak{C}$, hence $\mathbf{q} \triangleq \mathrm{vec}(\mathbf{G}_\mathfrak{C}) = \sum_{c\in\mathfrak{C}} \boldsymbol{\alpha}^\otimes(c)$. By defining the matrix $\mathbf{E} = \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{s}, \mathbf{I}) + \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{I}, \mathbf{s})$ which only depends on $\boldsymbol{\mathcal{T}}$ and $\mathbf{s}$, we can use the expression ${\boldsymbol{\alpha}^\otimes}^\top(c) = {\boldsymbol{\alpha}^\otimes}^\top \boldsymbol{\Xi}_{A^\otimes}(c)$ to see that:

$$\mathbf{q}^\top = \sum_{c\in\mathfrak{C}} (\boldsymbol{\alpha}^\otimes)^\top \boldsymbol{\Xi}_{A^\otimes}(c) = (\boldsymbol{\alpha}^\otimes)^\top \sum_{k\geq 0} \mathbf{E}^k$$
$$= (\boldsymbol{\alpha}^\otimes)^\top (\mathbf{I} - \mathbf{E})^{-1} \ ,$$

where we used the facts $\mathbf{E}^k = \sum_{c\in\mathfrak{C}:\mathrm{drop}(c)=k} \boldsymbol{\Xi}_{A^\otimes}(c)$ and $\rho(\mathbf{E}) < 1$ shown in the proof of Theorem 5.

Algorithm 2 summarizes the overall approximation procedure for the Gram matrices, which can be done to an arbitrary precision. There, reshape$(\cdot, n \times n)$ is an operation that takes an n^2-dimensional vector and returns the $n \times n$ matrix whose first column contains the first n entries in the vector and so on. Theoretical guarantees on the convergence rate of this algorithm are given in the following theorem.

Theorem 6. *There exists $0 < \rho < 1$ such that after k iterations in Algorithm 2, the approximations $\hat{\mathbf{G}}_\mathfrak{C}$ and $\hat{\mathbf{G}}_\mathfrak{T}$ satisfy $\|\mathbf{G}_\mathfrak{C} - \hat{\mathbf{G}}_\mathfrak{C}\|_F \leq \mathcal{O}(\rho^k)$ and $\|\mathbf{G}_\mathfrak{T} - \hat{\mathbf{G}}_\mathfrak{T}\|_F \leq \mathcal{O}(\rho^k)$.*

Proof. See supplementary material. □

Algorithm 2 `GramMatrices`

```
Input: A strongly convergent minimal WTA $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}\rangle$
Output: Gram matrices $\hat{\mathbf{G}}_\mathfrak{C} \simeq \sum_{c\in\mathfrak{C}} \boldsymbol{\alpha}_A(c)\boldsymbol{\alpha}_A(c)^\top$ and $\hat{\mathbf{G}}_\mathfrak{T} \simeq \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}_A(t)\boldsymbol{\omega}_A(t)^\top$
  Let $\boldsymbol{\mathcal{T}}^\otimes = \boldsymbol{\mathcal{T}} \otimes \boldsymbol{\mathcal{T}} \in \mathbb{R}^{n^2 \times n^2 \times n^2}$, and let $\boldsymbol{\omega}_\sigma^\otimes = \boldsymbol{\omega}_\sigma \otimes \boldsymbol{\omega}_\sigma \in \mathbb{R}^{n^2}$ for all $\sigma \in \Sigma$.
  Let $\mathbf{I}$ be the $n^2 \times n^2$ identity matrix and let $\mathbf{s} = \mathbf{0} \in \mathbb{R}^{n^2}$
  repeat
    $\mathbf{s} \leftarrow \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{s}, \mathbf{s}) + \sum_{\sigma\in\Sigma} \boldsymbol{\omega}_\sigma^\otimes$
  until convergence
  $\mathbf{q} = (\boldsymbol{\alpha} \otimes \boldsymbol{\alpha})^\top \left(\mathbf{I} - \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{I}, \mathbf{s}) - \boldsymbol{\mathcal{T}}^\otimes(\mathbf{I}, \mathbf{s}, \mathbf{I})\right)^{-1}$
  $\hat{\mathbf{G}}_\mathfrak{T} = \mathrm{reshape}(\mathbf{s}, n \times n)$
  $\hat{\mathbf{G}}_\mathfrak{C} = \mathrm{reshape}(\mathbf{q}, n \times n)$
  return $\hat{\mathbf{G}}_\mathfrak{C}, \hat{\mathbf{G}}_\mathfrak{T}$
```

4 Experiments

In this section, we assess the performance of our method on a model arising from real-world data, by using a PCFG learned from a text corpus as our initial model. Before presenting our experimental setup and results, we recall the standard mapping between WCFG and WTA.

4.1 Converting WCFG to WTA

A *weighted context-free grammar* (WCFG) in Chomsky normal form is a tuple $G = \langle \mathcal{N}, \Sigma, \mathcal{R}, \text{weight}\rangle$ where $\mathcal{N}$ is the finite set of nonterminal symbols, Σ is the finite set of words, with $\Sigma \cap \mathcal{N} = \emptyset$, $\mathcal{R}$ is a set of rules having the form $(a \to bc)$, $(a \to x)$ or $(\to a)$ for $a, b, c \in \mathcal{N}, x \in \Sigma$, and weight : $\mathcal{R} \to \mathbb{R}$ is the weight function which is extended to the set of all possible rules by letting weight$(\delta) = 0$ for all rules $\delta \notin \mathcal{R}$.

A WCFG G assigns a weight to each derivation tree τ of the grammar given by weight$(\tau) = \prod_{\delta\in\mathcal{R}} w(\delta)^{\sharp_\delta(\tau)}$ (where $\sharp_\delta(\tau)$ is the number of times the rule δ appears in τ), and it computes a function $f_G : \Sigma^+ \to \mathbb{R}$ defined by $f_G(w) = \sum_{\tau\in T(w)} \text{weight}(\tau)$ for any $w \in \Sigma^+$, where $T(w)$ is the set of trees deriving the word w.

Given a WCFG G, we can build a WTA that assigns to each binary tree $t \in \mathfrak{T}_\Sigma$ the sum of the weights of all derivation trees of G having the same topology as t. Let $G = \langle \mathcal{N}, \Sigma, \mathcal{R}, w\rangle$ be a WCFG in normal form with $\mathcal{N} = [n]$. Let $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}_{\sigma\in\Sigma}\rangle$ be the WTA with n states defined by $\boldsymbol{\alpha}(i) = \text{weight}(\to i)$ for all $i \in [n]$, $\boldsymbol{\mathcal{T}}(i, j, k) = \text{weight}(i \to jk)$ for all $i, j, k \in [n]$, and $\boldsymbol{\omega}_\sigma(i) = \text{weight}(i \to \sigma)$ for all $i \in [n]$, $\sigma \in \Sigma$. Then for all $w \in \Sigma^+$ we have $f_G(w) = \sum_{t\in\mathfrak{T}_\Sigma:\langle t\rangle=w} f_A(t)$. It is important to note that in this conversion the number of states in A corresponds to the number of nonterminals in G. A similar construction can be used to convert any WTA to a WCFG where each state in the WTA is mapped to a non-terminal in the WCFG.

4.2 Experimental Setup and Results

In our experiments, we used the annotated corpus of German newspaper texts NEGRA (Skut et al., 1997). We use a standard setup, in which the first 18,602 sentences are used as a training set, the next 1,000 sentences as a development set and the last 1,000 sentences as a test set S_{test}. All trees are binarized as described in (Cohen et al., 2013b). We extract a binary grammar in Chomsky normal form from the data, and then estimate its probabilities using maximum likelihood. The resulting PCFG has $n = 211$ nonterminals. We compare our method against the ones described in (Cohen et al., 2013a), who use tensor decomposition algorithms (Chi and Kolda, 2012) to decompose the tensors of an underlying PCFG.[2]

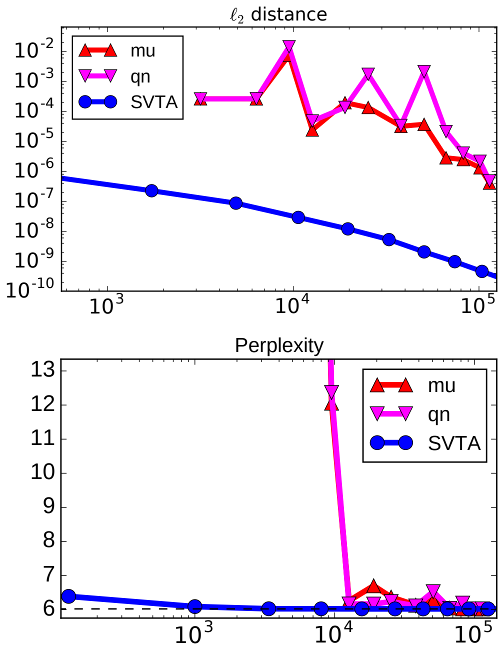

Figure 2: (top) ℓ_2 distance between functions. (bottom) Perplexity on the test set. The x axis denotes in both cases the number of parameters used by the approximation.

We used three evaluation measures: ℓ_2 distance (between the functions of type $\mathfrak{T}_\Sigma \to \mathbb{R}$ computed by the original WTA and the one computed by its approximation), perplexity on a test set, and parsing accuracy on a test set (comparing the tree topology of parses using the bracketing F-measure). Because the number of states on a WTA and the CP-rank of tensor decomposition method are not directly comparable, we plotted the results using the number of parameters needed to specify the model on the horizontal axis. This number is equal to $\hat{n}^3$ for a WTA with $\hat{n}$ states, and it is equal to $3Rn$ when the tensor $\boldsymbol{\mathcal{T}}$ is approximated with a tensor of CP-rank R (note in both cases these are the number of parameters needed to specify the tensor occurring in the model).

The ℓ_2 distance between the original function f and its minimization $\hat{f}$, $\|f - \hat{f}\|_2^2 = \sum_{t\in\mathfrak{T}}(f(t) - \hat{f}(t))^2$, can be approximated to an arbitrary precision using the Gram matrices of the corresponding WTA (which follows from observing that $(f - \hat{f})^2$ is rational). The perplexity of $\hat{f}$ is defined by $2^{-H_{\text{test}}}$, where $H_{\text{test}} = \sum_{t\in S_{\text{test}}} f(t) \log_2 \hat{f}(t)$ and both f and $\hat{f}$ have been normalized to sum to one over the test set. The results are plotted in Figure 2, where an horizontal dotted line represents the performance of the original model. We see that our method outperforms the tensor decomposition methods both in terms of ℓ_2 dis-

[2] We use two tensor decomposition algorithms from the tensor Matlab toolbox: `pqnr`, which makes use of projected quasi-Newton and `mu`, which uses a multiplicative update. See `http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html`.

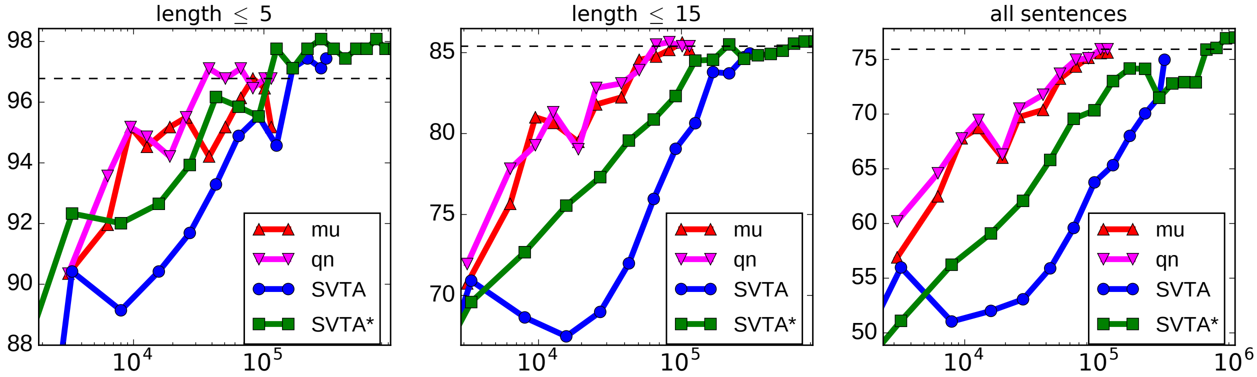

Figure 3: Parsing accuracy on the test set for different sentence lengths. The x-axis denote the number of parameters used by the approximation. The y-axis denotes bracketing accuracy.

tance and perplexity. We also remark that our method obtains very smooth curves, which comes from the fact that it does not suffer from local optima problems like the tensor decomposition methods.

For parsing we use minimum Bayes risk decoding, maximizing the sum of the marginals for the nonterminals in the grammar, essentially choosing the best tree topology given a string (Goodman, 1996). The results for various length of sentences are shown in Figure 3, where we see that our method does not perform as well as the tensor decomposition methods in terms of parsing accuracy on long sentences. In this figure, we also plotted the results for a slight modification of our method (SVTA*) that is able to achieve competitive performances. The SVTA* method gives more importance to long sentences in the minimization process. This is done by finding the highest constant $\gamma > 0$ such that the function $f_\gamma : t \mapsto \gamma^{\mathrm{size}(t)} f(t)$ is still strongly convergent. This function is then approximated by a low-rank WTA computing $\hat{f}_\gamma$, and we let $\hat{f} : t \mapsto \gamma^{-\mathrm{size}(t)} \hat{f}_\gamma(t)$ (which is rational). In our experiment, we used $\gamma = 2.4$. While the SVTA* method improved the parsing accuracy, it had no significant repercussion on the ℓ_2 and perplexity measures. We believe that the parsing accuracy of our method could be further improved. Seeking techniques that combines the benefits of SVTA and previous works is a promising direction.

Overall, the results are more promising for language modeling (ie. perplexity) than parsing. This is in line with a recent discovery on learning PCFGs (Scicluna and De La Higuera, 2014) showing that it is hard to perform simultaneously well on parsing and language modeling.

5 Conclusion

We described a technique for approximate minimization of WTA, yielding a model smaller than the original one which retains good approximation properties. Our main algorithm relies on a singular value decomposition of an infinite Hankel matrix induced by the WTA. Our experiments with real-world parsing data show that the minimized WTA, depending on the number of singular values used, approximates well the original WTA on three measures: perplexity, bracketing accuracy and ℓ_2 distance of the tree weights. Our work has connections with spectral learning techniques for WTA, and exhibits similar properties as those algorithms; e.g. absence of local optima. In future work we plan to investigate the applications of our approach to the design and analysis of improved spectral learning algorithms for WTA. We will also conduct a thorough theoretical analysis of the error induced by the truncation of an SVTA. On the practical side we want to investigate strategies to sparsify the SVTA obtained after minimization (or to maintain sparsity through the minimization process) in order to further improve the time complexity of WTA algorithms. Finally, we think that investigating the properties of the internal feature representations used by an SVTA might provide useful joint embeddings for the words, non-terminals, and productions rules arising from (L)PCFGs.

Acknowledgements

This work has been carried out thanks to the support of the French GIP ANR under contract ANR GRETA 12-BS02-004-01, the ARCHIMEDE Labex (ANR-11-LABX-0033), and the A*MIDEX project (ANR-11-IDEX-0001-02).

References

Bailly, R., Denis, F., and Ralaivola, L. (2009). Grammatical inference as a principal component analysis problem. In *Proceedings of ICML*.

Bailly, R., Habrard, A., and Denis, F. (2010). A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*.

Balle, B., Carreras, X., Luque, F., and Quattoni, A. (2014). Spectral learning of weighted automata: A forward-backward perspective. *Machine Learning*.

Balle, B., Panangaden, P., and Precup, D. (2015). A canonical form for weighted automata and applications to approximate minimization. In *Proceedings of LICS*.

Berstel, J. and Reutenauer, C. (1982). Recognizable formal power series on trees. *Theoretical Computer Science*.

Boots, B., Siddiqi, S., and Gordon, G. (2011). Closing the learning planning loop with predictive state representations. *International Journal of Robotics Research*.

Bozapalidis, S. and Louscou-Bozapalidou, O. (1983). The rank of a formal tree power series. *Theoretical Computer Science*.

Chi, E. C. and Kolda, T. G. (2012). On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*.

Cohen, S. B. and Collins, M. (2012). Tensor decomposition for fast parsing with latent-variable PCFGs. In *Proceedings of NIPS*.

Cohen, S. B., Satta, G., and Collins, M. (2013a). Approximate PCFG parsing using tensor decomposition. In *Proceedings of NAACL*.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013b). Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2014). Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *Journal of Machine Learning Research*.

Goodman, J. (1996). Parsing algorithms and metrics. In *Proceedings of ACL*.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*.

Kiefer, S., Marusic, I., and Worrell, J. (2015). *Minimisation of Multiplicity Tree Automata*.

Kulesza, A., Jiang, N., and Singh, S. (2015). Low-rank spectral learning with weighted loss functions. In *Proceedings of AISTATS*.

Kulesza, A., Rao, N. R., and Singh, S. (2014). Low-Rank Spectral Learning. In *Proceedings of AISTATS*.

Maletti, A. and Satta, G. (2009). Parsing algorithms based on tree automata. In *Proceedings of IWPT*.

Scicluna, J. and De La Higuera, C. (2014). Pcfg induction for unsupervised parsing and language modelling. In *Proceedings of EMNLP*.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Conference on Applied Natural Language Processing*.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.